

PRV

PATENT- OCH REGISTRERINGSVERKET
Patentavdelningen

10/510167
PCT/SE 03/00536

Intyg Certificate

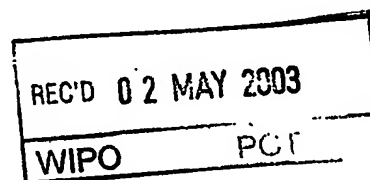
Härmed intygas att bifogade kopior överensstämmer med de handlingar som ursprungligen ingivits till Patent- och registreringsverket i nedannämnda ansökan.

This is to certify that the annexed is a true copy of the documents as originally filed with the Patent- and Registration Office in connection with the following patent application.

(71) Sökande Xelerated AB, Stockholm
Applicant (s)

(21) Patentansökningsnummer 0201020-5
Patent application number

(86) Ingivningsdatum 2002-04-04
Date of filing



Stockholm, 2003-04-16

För Patent- och registreringsverket
For the Patent- and Registration Office

Lina Oljeqvist
Lina Oljeqvist

Avgift
Fee

PRIORITY DOCUMENT
SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH
RULE 17.1(a) OR (b)

METHOD AND APPARATUS FOR PROCESSING DATA

Field of the invention

The present invention relates to data processing in general, and more particularly to a method and apparatus for pipelined processing of data.

5 **Background**

Many computers process data in a pipelined process. A processor which uses a pipelined processing technique and which receives a stream of data to be operated upon by the processor can be divided into segments referred to as processing stages, each processing stage being capable of executing operations on data. The processing stages make up a
10 pipeline. Several data packets may simultaneously be present in the processor pipeline, being operated upon by different processing stages, and being brought forward to the next processing stage in the pipeline as time flows. Each processing stage can execute operations on the data present in the processing stage. Upon each clock tick, data is passed onto the subsequent stage by loading registers that are located between the previous and the
15 subsequent processing stages. The data thus becomes present in the subsequent stage.

Summary

A problem to which the present invention relates is how to improve the possibilities of utilising pipelined processing of data.

20 This problem is addressed by a method of pipelined processing of a data packet in a processing means comprising at least one processing stage. The method is characterised by associating information reference to said data packet, said information reference comprising information relating to the length and position of information contained in the data packet. The method is further characterised in that, if said data packet is processed in a
25 processing stage in a manner so that the length and/or position of said information contained in the data packet is changed, then the information reference is altered in order to reflect said change.

30 The problem is further addressed by a processing means for pipelined processing of a data packet, and by an integrated circuit and a computer unit comprising said processing means.

The processing means comprises at least one processing stage comprising a logic unit and a register for storing at least part of said data packet. The processing means is characterised in that at least one register for storing information reference associated with said data packet is accessible to said logic unit, and at least one of at said at least one logic units is adapted to operate upon said information reference.

By the inventive method and processing means is achieved that the information contained in a data packet can be operated upon, by a pipelined processor, in a manner so that the length of the information contained in the data packet, and/or the position of the information in the data packet, is altered. By altering the value of the information reference accordingly upon such operations, information will always be available about the length and position of the information in the data packet.

In one embodiment of the invention, at least one bit is added to the data packet prior to associating information reference to the data packet. In this aspect of the invention, the processing means further comprises means for adding bits. Hereby is achieved that the information contained in the data packet when the data packet exits the processing means can occupy more bits than the number of bits that the data packet entering the processing means comprises. In this embodiment, the at least one bit is preferably added to the data packet in the beginning of the data packet as a dummy header, and/or at the end of the data packet as a dummy tail. Hereby is achieved that the method and processing means are made suitable for processing of data packets in a communication system in which headers and tails are added and removed from a data packet as the data packet is transmitted within the communication system. The means for adding bits could suitably comprise a buffer and a shifter. Advantageously, the shifter could be a barrel shifter. Hereby is achieved that the number of bits being added to a data packet is flexible. The number of bits being added could e.g. differ between each packet, be static, or be varied from time to time according to the desire of the operator of the processing means.

In one aspect of the invention, at least one bit is removed from the data packet upon the data packet exiting the last one of the processing stages. In this aspect of the invention, the inventive processing means further comprises means for removing at least one bit from said data packet. Hereby is achieved that the use of bandwidth is made efficient, and that

bits not containing any information can be removed. Preferably it is determined, prior to the removal of bits, whether any bits of the data packet are superfluous, and if so, then said superfluous bits are removed. Hereby is achieved that the use of bandwidth is optimised. The means for removing bits could suitably comprise a shifter and buffer. Said shifter

5 could advantageously be a barrel shifter. The barrel shifter could use the information reference to determine how the bits of the data packet should be shifted.

The information reference could preferably be included in additional information associated with said data packet. The at least one processing stage of said processing means

10 could then comprise at least one register for storing information reference. Hereby is achieved that processing of information reference can be made fast, and, when the data packet is divided into at least two data blocks, that the information reference can slide backwards and/or forwards within the data blocks in order to be available only to the processing stage operating on either one of the data blocks.

15 ***Brief description of the drawings***

Fig. 1 shows an example of a data packet.

Fig. 2 is a schematic illustration of a processing pipeline comprising two processing stages.

20 Fig. 3a-d illustrate how a data packet is operated upon according to an embodiment of the invention.

Fig. 4a-d illustrate how a data packet is operated upon according to another embodiment of the invention.

25 Fig. 5 is a flowchart, schematically illustrating an embodiment of the inventive method.

Fig. 6a-d illustrate an embodiment of how information reference can slide backwards within a set of data blocks in a processing pipeline.

30 Fig. 7 illustrates processing means according to an embodiment of the invention.

Detailed description

Most data communication systems consist of a number of nodes, in which data may be processed and between which nodes data packets are transmitted by use of several protocols. A node can use one or more protocols for the transmission of data packets.

5 When a data packet is transmitted using a protocol, the transmitting node may add a protocol header and/or a protocol tail to the data packet in order to add information necessary to the further transmission of the data packet. Similarly, when a node receives a data packet, the receiving node may remove a protocol header and/or a protocol tail from the data packet in order to unpack the data contained in the data packet. A typical data
10 packet 100 is shown in Fig. 1, where user data 110 is encapsulated in a header 120 added by a first protocol, a header 130 and a tail 140 added by a second protocol, and a header 150 added by a third protocol. As the data packet 100 is transmitted within the communication network, nodes will repeatedly encapsulate the data packet 100 by adding headers and/or tails, and decapsulate data packet 100 by removing headers and/or tails.

15

In Fig. 2, an example of a processing pipeline 200 comprising two processing stages 205a and 205b is shown. Obviously, a pipeline 200 may comprise more than two processing stages 205. The processing stages 205a and 205b comprise logic units 210a and 210b, respectively, in which the operation on data is performed. A data block 215, comprising
20 one or several data packets 100, or parts of a data packet 100, is stored in a data block register 220a upon entering the pipeline 200. Additional information 225 associated with data block 215, such as e.g. information about which instructions should be executed on data block 215 in pipeline 200 (see Swedish patent application 0100221-1, filed by the applicant and hereby incorporated by reference), may accompany the data block 215 and
25 can be stored in one or more additional registers 230a. The additional information 225, as well as the data blocks 215, may or may not be operated upon in the pipeline 200. As data block 215 and additional information 225 has entered the pipeline, they will be processed by logic unit 210a. Upon a first clock tick, the data block 215 and additional information 225 will be stored in data block register 220b and additional register 230b, respectively,
30 which registers are accessible by processing stage 205a and processing stage 205b. A second data block (not shown in the figure) may then enter register 220a, possibly accompanied by associated additional information that could enter register 230a. Upon a second clock tick, data block 215 and additional information 225 will be present in

processing stage 205b, while the second data block will be present in processing stage 205a. Upon a third clock tick, data block 215 and additional information 225 will be stored in registers 220c and 230c, respectively, while the second data block will be stored in data block register 220b. A third data block 215 may now enter the pipeline, to be stored in data block register 220a.

A register for storing data, such as registers 220 and 230, can only store up to a pre-determined maximum of data bits, and a processing stage 205 can only process a pre-determined amount of data bits at a time. When adjusting the flow of data through pipeline 200, these limits of the registers 220, 230 and the processing stages 205 would have to be considered. A further consequence hereof is that in a pipelined processing environment, to add bits from the data present in a pipeline 200 would cause huge problems in terms of e.g. interference with preceding and/or following data blocks 215. Therefore, designing pipelined processors or ASICs to be used in systems in which bits are added and/or removed by the processors, such as e.g. data communication systems where headers and/or tails are regularly added to, and removed from, data packets, is not a straightforward process. In order to allow for changing the size of data packets within the pipeline 200, complex logic for dynamically shifting data at any stage of pipeline 200 would be required, as well as some flexible queuing of data blocks 215.

A solution to the problem of how to be able to vary the number of bits of a data packet 100 that is processed in a pipeline 200 is to add dummy bits to the data packet 100, prior to the data packet 100 entering the pipeline 200. Variable(s) for recording the length of the data packet 100 (*i.e.* the number of bits contained in data packet 100), and the position of the first bit of data packet 100, could then be associated with data packet 100. As data packet 100 is operated upon and the length and position of the information contained in data packet 100 is altered, these variable(s) of recording could be altered accordingly.

In the following, it will be assumed that the number of bits added to a data packet 100 is a multiple of eight, *i.e.* the added bits can easily be formed into bytes. However, any number of bits could be added to a data packet 100.

Fig. 3 schematically illustrates a received data packet 100 being encapsulated according to an embodiment of the inventive method. In Fig. 3a, a received data packet 100 comprising n bytes is shown. In Fig. 3b, a dummy header comprising m bytes is added to the received data packet 100, as well as a dummy tail 310 comprising k bytes, the received data packet 100, the dummy header 305 and the dummy tail 310 making up an intermediate packet 315. The shaded colour of dummy header 305 and dummy tail 310 indicates that the bytes contained therein do not represent any information, *i.e.* dummy header 305 and dummy tail 310 are empty. Additional information 225 comprising information about the length of the information contained in intermediate packet 315, as well as about the position of the information contained in the intermediate packet 315, hereinafter referred to as information reference 320, could then be associated with intermediate packet 315. When intermediate packet 315 is first generated, information reference 320 should preferably contain information about the length of the received data packet 100, as well as information about the position of received data packet 100 within intermediate packet 315. In Fig. 3, such information reference 320 is illustrated by a length value 325, indicating the length of the part of the intermediate packet 315 that contains information, and an offset value 330, indicating the position of the first byte of intermediate packet 315 that contains information. The length value 325 and the offset value 315 could preferably be stored in different additional registers 230. In Fig. 3b, the length value 325 takes the value n , while offset value 330 takes the value m . In other implementations, such information reference 320 could comprise information representing the position of the first byte of information and the last byte of information in intermediate packet 315, or information representing the length of the information contained in intermediate packet 315 and the position of the last byte of information.

Fig. 3c illustrates that the intermediate packet 315 has been executed upon by one or several of the logic units 210 in pipeline 200. Parts of the dummy header 305 and the dummy tail 310 of the intermediate packet 315 of Fig. 3c are used to represent information, so that the size of the data containing information is increased from n bytes to l bytes, where $l \leq n + m + k$. In the example given in Fig. 3c, p bytes of dummy header 305 and q bytes of dummy tail 310 are used for information. This is illustrated by the shaded area of intermediate packet 315 being smaller than the shaded area of intermediate packet 315 shown in Fig. 3b, a shaded area representing empty bytes. Accordingly, the value of length

value 325 of Fig. 3c is $n+p+q$, while the value of the offset value 330 is $m-p$. An example of an operation that would result in this scenario could be the encapsulation of a data packet 100 by a transmitting node in a data communication system by adding a header and a tail comprising information relevant to the following transmission of the data packet 100.

5 Another example could be a local subsystem, used for transmission of data to another local subsystem within the same node, encapsulating data according to a local subsystem protocol. Yet another example is the encapsulation of data according to a local hardware protocol, used for the transmission of data between hardware components on a hardware board, or between hardware boards.

10

In Fig. 3d, the bytes that are still not representing any information has been removed, yielding a resulting data packet 100 comprising more bytes than the received data packet 100. Assuming that no operation that has resulted in changes of the length of the information stored in the intermediate packet 315 has been performed, other than the changes indicated in Fig. 3b, the amount of bytes to be removed is $m-p$ at the header end of intermediate packet 315, and $k-q$ at the tail end of the intermediate packet 315.

15

Alternatively, some or all of the bytes in intermediate packet 315 that are empty could be kept as part of resulting data packet 100. The removal of the superfluous bytes of intermediate packet 315 could advantageously be performed after the packet has exited the pipeline 200. Alternatively, the removal could be performed at the end of the pipeline 200.

20

Naturally, rather than using only some of the bytes in the dummy header 305 and the dummy tail 310, all bytes of dummy header 305 and dummy tail 310 could be used for the representation of information. There would then be no superfluous bytes to remove, and the resulting data packet 100 would be the same as the intermediate packet 315 that exits the last processing stage of pipeline 200. A scenario could also occur where bytes from only one of the dummy header 305 or the dummy tail 310 have been used for representing information when the intermediate packet 315 leaves the pipeline 200. In some instances, it may occur that none of the bytes in dummy header 305 or dummy tail 310 are used for representing information. The inventive method could advantageously be used also for a situation where the resulting data packet 100 contains less information than the received data packet 100. Obviously, any combination of adding/removing information at the header/tail end of the received data packet 100 can be performed by the inventive method.

25

30

The decapsulation of a received data packet 100 according to an embodiment of the present invention is illustrated in Fig. 4. Fig. 4a shows a received data packet 100 comprising n bytes of information. Fig. 4b corresponds to Fig. 3b, where a dummy header 305 containing m bytes and a dummy tail 310 comprising k bytes are added to the received data packet 100, resulting in an intermediate packet 315. Fig. 4c illustrates that the intermediate packet 315 has been operated upon by at least one of the logic units 210 of pipeline 200. Not all of the information contained in the received data packet 100 is still useful, and the amount of bytes representing empty information has increased compared to the intermediate packet 315 that was initially generated. In the example given in Fig. 4c, the length of the information contained in intermediate packet 315 has been reduced by r bytes at the header end of the intermediate packet 315, and by s bytes at the tail end. The values of the length value 325 and the offset value 330 have accordingly been changed into $n-r-s$ and $m+r$, respectively. An example of an operation that would result in this scenario is the unpacking of a data packet 100 by a receiving node in a communications system, where header(s) and/or tail(s) comprising information that was relevant only at previous stages of the transmission session are removed.

In Fig. 4d, the superfluous bytes of the intermediate packet 315 that exits the last processing stage 215 of pipeline 200 have been removed, resulting in a resulting data packet 100 comprising less bytes than received data packet 100.

In an embodiment of the invention where all data packets 100 processed by a pipeline 200 are decapsulated rather than encapsulated, so that the size of a received data packet 100 is always greater than the corresponding resulting data packet 100, the addition of bytes to the received data packet 100, illustrated by Figs. 3b and 4b, could be omitted. However, the information reference 320, also illustrated by Figs 3b and 4b, should preferably be generated even in such an embodiment.

The step illustrated by Figs. 3b and 4b could advantageously be performed prior to the received data packet 100 entering the pipeline 200, while the step illustrated by Figs. 3d and 4d could advantageously be performed after the intermediate packet 315 has exited the pipeline 200.

The number of bytes added to a received data packet 100 in order to form an intermediate packet 315 can vary from time to time. Each data packet 100 to be processed by a pipeline 200 could e.g. be associated with information about how many bytes should be added to the received data packet 100.

A flowchart describing an embodiment of the inventive method is schematically illustrated in Fig. 5. In step 500, a data packet 100 to be processed in a pipeline 200 enters the pipeline receiver, which could e.g. be positioned before the registers 220a and 230a of Fig. 2. In step 505, an intermediate packet 315 is created by increasing the size of received data packet 100 via adding to received data packet 100 additional bytes, either in form of a dummy header 305, a dummy tail 310, or both. In step 510, information reference 320 is created and associated with intermediate packet 315. Preferably, this information reference 320 is part of the additional information 225. The information reference 320 could e.g. be realised by a length value 325 and an offset value 330, cf. Figs. 3 and 4. Length value 325 and offset value 330 could preferably be stored in separate additional registers 230. In step 515, the intermediate packet 315 and the additional information 225 enter the pipeline 200, so that at least part of intermediate packet 315 and the additional information 325 is available for at least one of the processing stages 205 of pipeline 200. In step 520, at least one processing stage of 205 processes at least part of intermediate packet 315. In step 525, it is checked whether any operation performed on intermediate packet 315 in step 525 results in that information reference 320 should be changed. If so, step 530 is entered, where the information reference 320 is changed accordingly. Then step 535 is entered. If in step 525 it is found that no changes to the information reference 320 is necessary, then step 535 is entered directly. In step 535 it is checked whether any further processing of intermediate packet 315 will take place, i.e. if there will be any parts of intermediate packet 315 present in any of the processing stages 205 upon the next clock. If so, then step 520 is re-entered, so that for each clock tick upon which at least part of the intermediate packet 315 is available for processing by at least one of the processing stages 205, the loop made up of step 520, 525, 535 and, where applicable, step 530, is run. If it is found in step 535 that no more processing of intermediate packet 315 will take place in pipeline 200, then step 540 is entered, in which step it is checked whether any bytes should be removed from intermediate packet 315. This could preferably be performed by way of checking the value

of information reference 320. If any bytes should be removed from intermediate packet 315, then step 545 is entered, in which superfluous bytes at the header end and/or the tail end of intermediate packet 315 is removed according to the value of information reference 320. Step 550, where the data packet 100 exits the pipeline, is then entered. If in step 540 it is found that no superfluous bytes should be removed, then step 550 is entered directly.

The flowchart of Fig. 5 could be altered in many ways without departing from the spirit of the invention. For example, step 540 in which it is checked whether any bytes should be removed from intermediate packet 315 could be omitted, and step 545 entered directly after step 535. Alternatively, steps 540 and 545 could be omitted, data packet 100 exiting the pipeline in step 550 then including any superfluous bytes. Furthermore, step 525 could e.g. be implemented so that the program executing on intermediate packet 315 in step 520 also executes changes to the information reference 320, in conjunction with executing the changes of intermediate packet 315 giving rise to the need of changing the length value 325 and offset value 330. Step 525 could then be omitted. Alternatively, a flag could be set in step 520, indicating whether or not the length and position of the information contained in intermediate packet 315 has been changed, and step 525 would then comprise checking the value of said flag. As discussed above in relation to Figs. 3 and 4, the step 505 could advantageously be omitted in embodiments of the invention in which all received data packets 100 will be decapsulated by pipeline 200.

Depending of the size of a received data packet 100 and the bandwidth of pipeline 200, the received data packet 100 may have to be divided into two or more data blocks 215. The size of a data block 215 is a question of implementation, and any size of data block 215 could be used. In one embodiment of the application, given by way of example, the size of a data block 215 is 64 bytes. A received data packet 100 containing 150 bytes of information would in this embodiment be divided into at least 3 blocks of 64 bytes each, making up an intermediate packet 315. In the case of 150 bytes being divided upon 3 data blocks of 64 bytes each, the intermediate packet 315 contains 192 bytes, of which 42 bytes can be distributed amongst a dummy header 305 and/or a dummy tail 310. If more extra bytes are desired, additional, empty, blocks could optionally be added to the intermediate packet 315, yielding a larger dummy header 305 and/or dummy tail 310. Alternatively,

additional bytes can be added to received data packet 100, in order to form an intermediate packet 315, before the intermediate packet 315 is divided into data blocks 215.

When an intermediate packet 315 is divided into data blocks 215 and each data block 215
5 is operated upon separately by the processing stages 205 of pipeline 200, only one
information reference 320 should preferably be associated with the group of data blocks
215 representing the intermediate packet 315. When an intermediate packet 315 enters a
pipeline 200, the information reference 320 should preferably enter the pipeline together
10 with the data block 215 that enters the pipeline 200 first (*cf.* additional information 225
accompanying data block 215 in Fig. 2). As other data blocks 215 enter the pipeline 200,
operations that give rise to the necessity of altering the information reference 320 may be
performed on any data block 215 of intermediate packet 315 present in any of the
processing stages 205 of pipeline 200. Hence, the information reference 320 should
15 advantageously be available to the logic unit 210 of the processing stage 205 in which such
operations are performed, at the time of the operations being performed, in order to provide
for the possibility of keeping the information reference 320 up to date at all times.

Figs. 6a-d illustrate the flow of an intermediate packet 315A through a pipeline 200
according to an embodiment of the invention. The intermediate packet 315A is divided into
20 two data blocks 215, referred to as data blocks 215 A0 and 215 A1, and accompanied by
information reference 320 A. Intermediate packet 315A may comprise a dummy header
305 and/or a dummy tail 310. Additional information 225 other than information reference
320 may accompany intermediate packet 315A, or each individual data block 215 A0-A1,
but to simplify the description, this other additional information 225 is not illustrated in
Fig. 6.
25

The pipeline 200 of Figs. 6a-d consists of three processing stages 205a-c, each comprising
a logic unit 210, referred to as logic unit 210a-c, respectively. It should be understood that
pipeline 200 may comprise any number of processing stages 205. To a logic unit 210a-c,
30 one or more operations for operating on data blocks 215 are available, as is illustrated by
each logic unit 210 in the Figs. 6a-d containing a sequence 600a-c of a flow diagram. To
simplify the description, only one operation is available to each logic unit 210 of Fig. 6,
30

although it should be understood that more complicated structures of operations may be implemented.

As time flows, each data block 215 proceeds through the pipeline 200, so that each data block 215 is available for processing in each logic unit 210 during a period of time corresponding to the time that passes between two consecutive clock ticks. Each processing stage 205 of Fig. 6 comprises a data block register 220 and an additional register 230. During the time when a certain data block 215 is available to a certain logic unit 210, logic unit 210 may or may not operate on the data block 215. If the logic unit 210 operates on data block 215, the additional information 225, such as information reference 320, associated with the intermediate packet 315 which data block 215 is a part of, should preferably be available for processing by the logic unit 215.

Figs. 6a-d each illustrate a separate period of time, each period of time corresponding to the time interval that passes between two consecutive clock tick ticks. Fig. 6a illustrates a first clock tick in which the first data block 215 A0 of intermediate packet 315A has entered the first processing stage 205a of pipeline 200. Information reference 320A has also entered processing stage 205a, and data block 215 A0 and information reference 320A are stored in data block register 220a and additional register 230a, respectively. As illustrated by the bold line around sequence 600a, logic unit 205a operates on data block 215 A0 during this clock tick. If the length of the information contained in intermediate packet 315, or the position of said information, are affected by the operation performed on data block 215 A0, then logic unit 210a operates on information reference 320A in order for information reference 320A to reflect these changes.

In Fig. 6b, a second clock tick is illustrated, in which data block 215 A0 and information reference 320A have been forwarded to processing stage 210b. Data block 215 A1 has entered processing stage 210a. Since data block 215 A2 is not accompanied by any information reference 320 at this point in time, additional register 230a is empty (or contains non-useful information). Logic unit 210b operates on data block 215 A0, as is indicated by the bold line around sequence 600b. Logic unit 210a, on the other hand, does not operate on data block 215 A1. By the operation performed on data block 215 A0 by logic unit 210b, it is determined whether information reference 320A should be moved

forward to additional register 230c during the next clock tick. As is illustrated by the "false" way out of sequence 600b being bold, it is determined that information reference 320 A should not be moved forward, but it should rather slide backwards within the set of data blocks 215 making up intermediate packet 315. This indicates that during the next clock tick, data block 215 A1 will be operated upon, rather than data block 215 A0.

In accordance with the operation illustrated in Fig. 6b, Fig. 6c, illustrating a third clock tick, shows a situation where data blocks 215 A0 and 215 A1 are each moved forward to the next processing stages 205c and 205b, respectively, while information reference 320A remains in additional register 230b. Data block 215 A2 has entered processing stage 205a. During the clock tick illustrated by Fig. 6c, logic unit 210b operates on data block 215 A1 in order to determine whether information reference 320b should move forward to processing stage 205c during the next clock tick. As is illustrated in the figure (the "1" of sequence 600b being bold), the result of the operation is that information reference 320b should be moved forward to processing stage 205c during next clock tick.

In Fig. 6d, a fourth clock tick is illustrated, in which data block 215 A0 has left pipeline 200, and data block 215 A1 and information reference 320A are stored in data block register 220c and additional register 230c, respectively. As is indicated by the bold line around sequence 600c, logic unit 210c performs an operation on data block 215 A1 during this clock tick. Should the operation performed on data block 215 A1 by logic unit 210c have altered the length or the position of the information contained in intermediate packet 315, then the information reference 320A is altered accordingly.

The process of sliding the information reference 320 backwards within the set of data blocks 215 that make up intermediate packet 315 is very efficient for providing the information reference 320 to a processing stage 205 that is positioned closer to the input of pipeline 200 than the processing stage 205 that last processed the additional information 320. However, in some cases, it might be necessary to slide the information reference 320 forwards within the set of data blocks 215, so that the information reference 320 can be operated upon by a processing stage 205 which is further away from the input of pipeline 200 than the processing stage 205 that last operated upon information reference 320. One way of sliding the information reference 320 forwards is to have synchronisation buffers at

different points in pipeline 200. To slide the information reference 320 forwards can e.g. be interesting when the intermediate packet 315 exits pipeline 200, in order to allow for the first byte of intermediate packet 315 to be accompanied by the information reference 320. A synchronisation buffer could then be positioned after the last processing stage 205 of pipeline 200.

The process of sliding additional information 225 backwards and forwards in the set of data blocks 215 forming an intermediate packet 315 is further described in the International patent application PCT/SE01/01133, filed by the applicant and hereby incorporated by reference.

As an alternative to implementing information reference 320 as part of additional information 225, information reference 320 could be stored in a separate memory available to all processing stages 205.

In Fig. 7, an example of a processing means 700 adapted to process data packets according to the inventive method is shown. Processing means 700 of Fig. 7 comprises a receiver 705, a pipeline 200 and a transmitter 710. The input of receiver 705 is connected to incoming line 707, and the output of receiver 705 is connected to pipeline 200. The pipeline 200 comprises a number of processing stages 205, cf. Fig. 2 and 6, and is further connected to the input of transmitter 710, the transmitter 710 being further connected, on its output side, to outgoing line 712. The incoming line 707, the pipeline 200 and the outgoing line 712 may each have different effective bandwidths, i.e. the speed at which data may be transmitted may differ between incoming line 707, pipeline 200 and outgoing line 712. The effective bandwidth of inventive pipeline 200 is greater than or equal to the effective bandwidths of incoming line 707 and outgoing line 712.

The receiver 705 is adapted to receive data packets 100 that are to be processed in pipeline 200. Receiver 705 comprises means 715 for adding bits to received data packets 100. Thus, intermediate packets 315 are generated in receiver 705. The means 715 for adding bits could e.g. comprise a receiver buffer 720 in which the bits of data packet 100 are stored upon reception, and a receiver shifter 725, to which the bits are forwarded from the receiver buffer 720. Receiver buffer 720, which is preferably a FIFO (First In First Out)

buffer, provides for the transition between the effective bandwidth of incoming line 707 and the effective bandwidth of pipeline 200. In receiver shifter 725, the bits are shifted according to how many dummy bits are desired in the dummy header 305 and the dummy tail 310 of intermediate packet 315, and the desired amount of additional bits are added.

5 Preferably, receiver shifter 725 could be a barrel shifter in which the shift performed by receiver shifter 725 can be varied. Alternatively, receiver shifter 725 could be a static shifter.

10 The transmitter 710 is adapted to transmit resulting data packets 100. Preferably, transmitter 710 comprises means 730 for removing bits from intermediate packets 315. The means 730 for removing bits could e.g. comprise a transmitter shifter 735 and a transmitter buffer 740. In transceiver shifter 725, the bits are shifted according to how many dummy bits should be removed in the dummy header 305 and the dummy tail 310 of intermediate packet 315, and the superfluous bits are removed. Transmitter shifter 735 could
15 advantageously be a barrel shifter, which could use the information of information reference 320 as input. Alternatively, transmitter shifter 735 could be a static shifter. Transmitter buffer 740, which preferably could be a FIFO buffer, provides for the transition between the effective bandwidth of pipeline 200 and the effective bandwidth of outgoing line 712.

20 In an embodiment of the inventive processing means that is to be used in an environment where the effective bandwidth of the pipeline 200 corresponds to the effective bandwidth of incoming line 707 plus the flow of additional bits added by means 715 for adding bits, then receiver buffer 720 could be omitted from processing means 700. Similarly, if the
25 effective bandwidth of outgoing line 712 corresponds to the effective bandwidth of pipeline 200 minus the flow of the bits that are removed means 730 for removing bits, then the transmitter buffer 740 could be omitted.

30 When dimensioning the receiver buffer 720, the relation between the effective bandwidths of incoming line 707 and pipeline 200 should be considered. The expected flow of data packets 100 on incoming line 707 could also be taken into account, as well as the expected size of the data packets 100. In an embodiment of the invention in which the amount of bits added by means 715 for adding bits can be varied on a data packet basis, the receiver buffer

720 could cater for storage of data packets 100 that demand an addition of bits, which, if a continuous stream of data packets 100 demanding the addition of that same amount of bits, would correspond to a higher effective bandwidth than the effective bandwidth of pipeline 200, provided that the average amount of bits added to incoming data packets 100 does not
5 yield a data flow that exceeds the effective bandwidth of pipeline 200. In a similar manner, when dimensioning the transmitter buffer 745, the relation between the effective bandwidths of pipeline 200 and outgoing line 712 should be accounted for.

The processing means 700 could be implemented as an integrated circuit (i.e. as an ASIC),
10 as part of an integrated circuit, or as many integrated circuits connected to each other.

The present invention could advantageously be implemented in any node in a data communication system, in which node data packets are processed so that the length or position of information contained in data packets are altered. Examples of such nodes are
15 routers and telecommunication switches for packet data. A processing means 700 could then be part of a computer unit, such as a network computer unit or a signal processing computer unit.

One skilled in the art will appreciate that the present invention is not limited to the
20 embodiments disclosed in the accompanying drawings and the foregoing detailed description, which are presented for purposes of illustration only, but it can be implemented in a number of different ways, and it is defined by the following claims.

CLAIMS

1. A method of pipelined processing of a data packet (100, 315) in a processing means (700) comprising at least two processing stages (205), said data packet (100) containing information, said method characterised by

5 associating (510) information reference (320; 325, 330) to said data packet (315), said information reference (320; 325, 330) comprising information relating to the length and position of the information contained in said data packet (315);
processing (520) said data packet (315) in a processing stage (205); and
10 if said processing (520) of said data packet (315) results in a change of the length or position of said information contained in said data packet (315), then altering (530) said information reference (320; 325, 330) in order for said information reference (320; 325, 330) to reflect said change.

2. The method of claim 1, further comprising the step of:

15 adding (505), prior to the step of associating (510), at least one bit (305, 310) to said data packet (100).

3. The method of claim 2, wherein

20 said step of adding (505) comprises adding a header (305) and/or a tail (310) to said data packet (100).

4. The method of any one of the above claims, the method further comprising the steps of:

25 determining, upon the data packet exiting the last of said at least one processing stages, (540) whether any bits of the data packet (315) are superfluous; and, if any bits of the data packet (315) are superfluous, then
removing (545) said superfluous bits.

5. The method of any one of claims 1-3, the method further comprising the steps of:

30 removing, upon the data packet exiting the last of said at least one processing stages, at least one bit from the data packet.

6. The method of any one of the above claims, wherein

said information reference (320) is included in additional information (225) associated with said data packet.

7. The method of any one of the above claims, wherein

5 prior to said step of processing (520) said data packet (315), said information reference (320) is stored in at least one register (230) accessible to the processing stage (205) performing said processing (520).

8. The method of any of the above claims, wherein

10 said information reference comprises a length value (325) and an offset value (330), said length value (325) representing the length of the information contained in said data packet (315) and said offset value (330) indicating the position in said data packet (315) of the information contained in said data packet (315).

15 9. A processing means for pipelined processing of a data packet (100, 315), said processing means comprising at least one processing stage comprising a logic unit (210) and a register (220) for storing at least part of said data packet (100, 315), said processing means being characterised in that

20 at least one register (230) for storing information reference (320) associated with said data packet (315) is accessible to said logic unit (210); and

at least one of at said at least one logic units (210) is adapted to operate upon said information reference (320).

10. The processing means of claim 9, further comprising

25 means (715) for adding at least one bit to said data packet (100).

11. The processing means of claim 10, wherein

said means (715) for adding comprises a buffer (720) and a shifter (725).

30 12. The processing means of any one of claims 9-11, the processing means further comprising

means (730) for removing at least one bit from said data packet (315).

13. The processing means of any one of claims 9-12, wherein

means (730) for removing comprises a shifter (735) and a buffer (740).

14. The processing means of claim 11 or claim 13, wherein

5 said shifter (725, 735) is a barrel shifter.

15. The processing means (700) of any one of claims 9-14, wherein

said at least one register (230) for storing information reference (230) is located in
said processing stage (205).

10

16. The processing means (700) of any of claims 9-15, wherein

said at least one register (230) for storing information reference comprises one
register (230) for storing a length value (325) and another register (230) for storing an
offset value (330).

15

17. An integrated circuit, **characterised** by

a processing means (700) according to claim 9.

18. A computer unit **characterised** by

20 an integrated circuit according to claim 9.

9
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20

Abstract

The present invention relates to a method and apparatus for pipelined processing of data. When a data packet containing information is received by a processor operating according to pipelined processing, bits are added to the data packet and an intermediate packet, comprising more bits than the received data packet, is generated. To the intermediate packet is associated information reference, the information reference comprising information regarding the length and position of the information in the intermediate packet. As the intermediate packet is processed, changes to the intermediate packet resulting in changes of the length or the position of the information in the intermediate packet will trigger changes of the information reference. When the intermediate packet exits the processor, superfluous bits are removed.

3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
22

Fig. 1

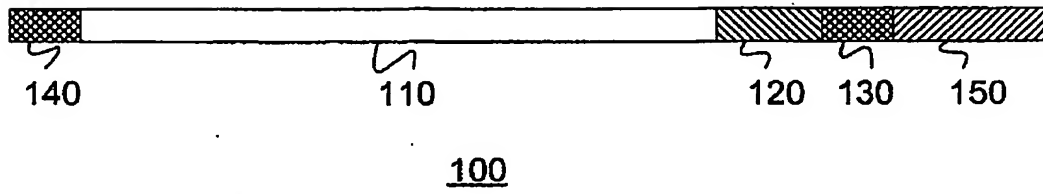


Fig. 2

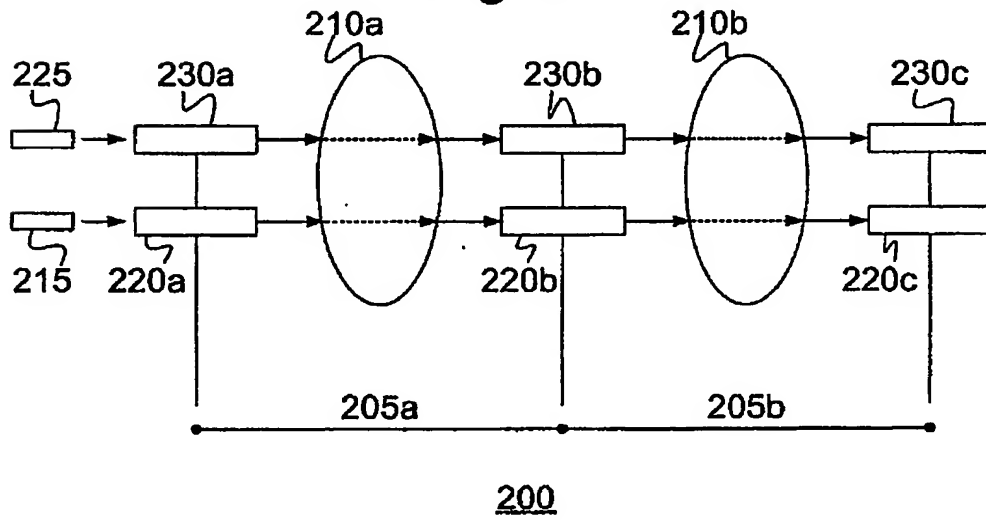
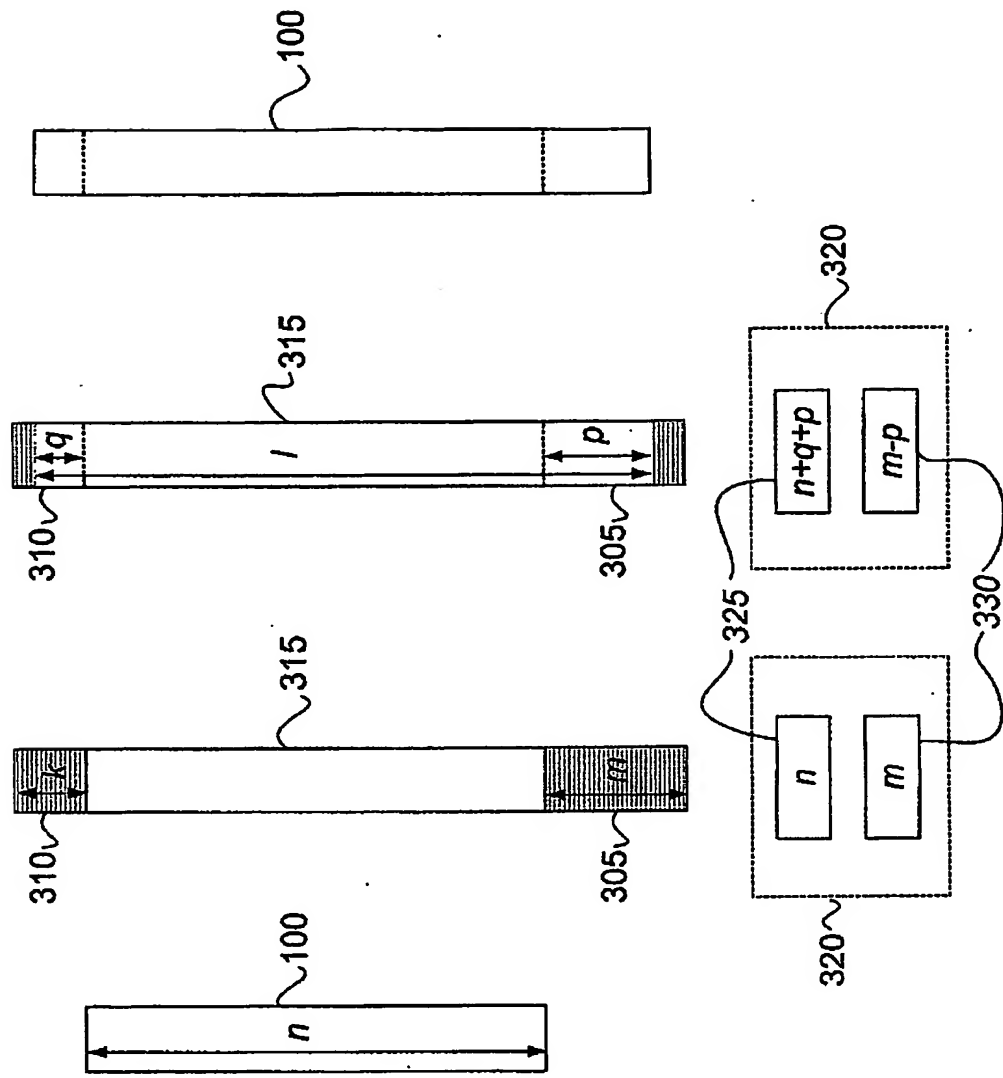


Fig. 3a Fig. 3b Fig. 3c Fig. 3d



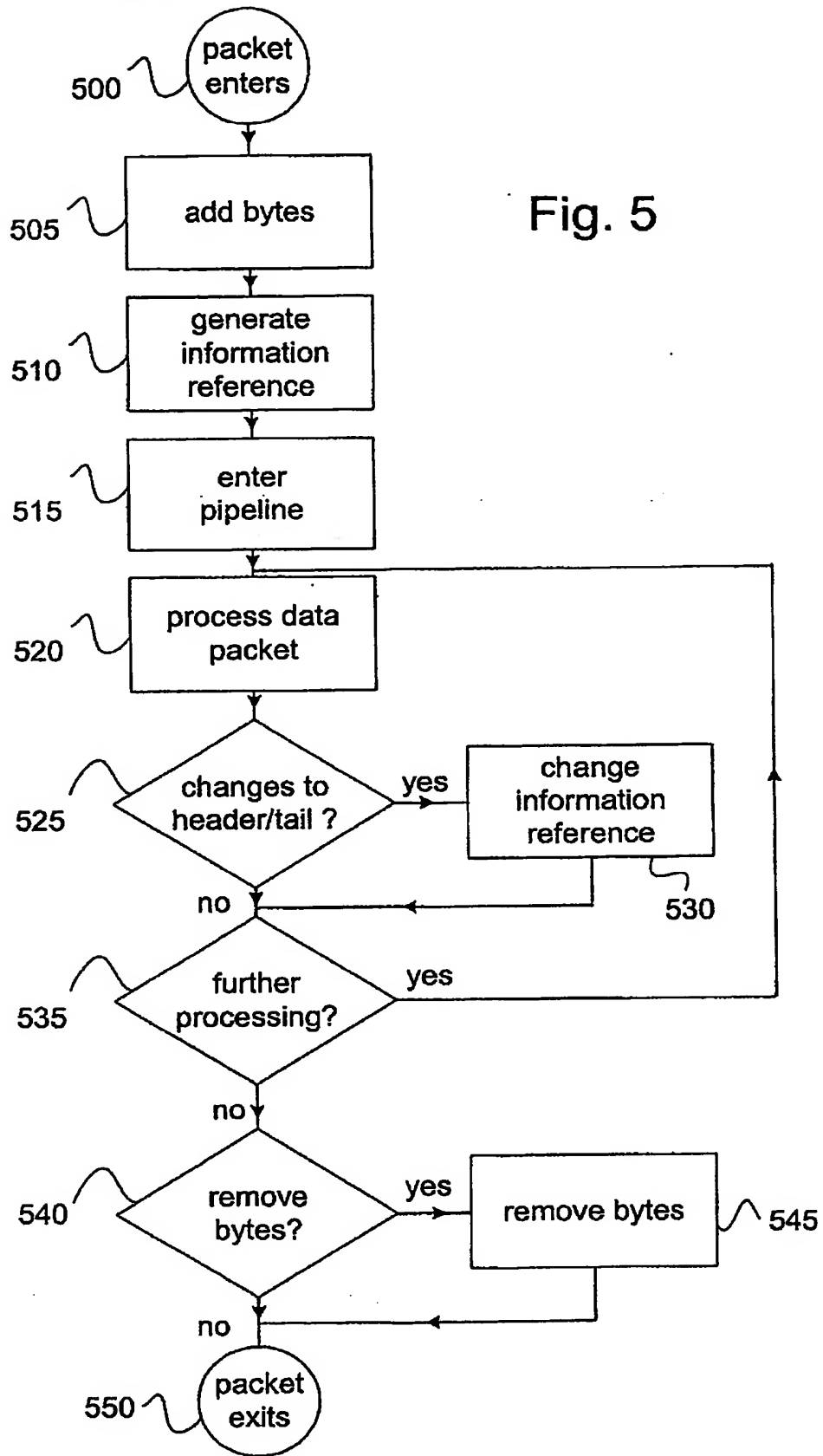


Fig. 6a

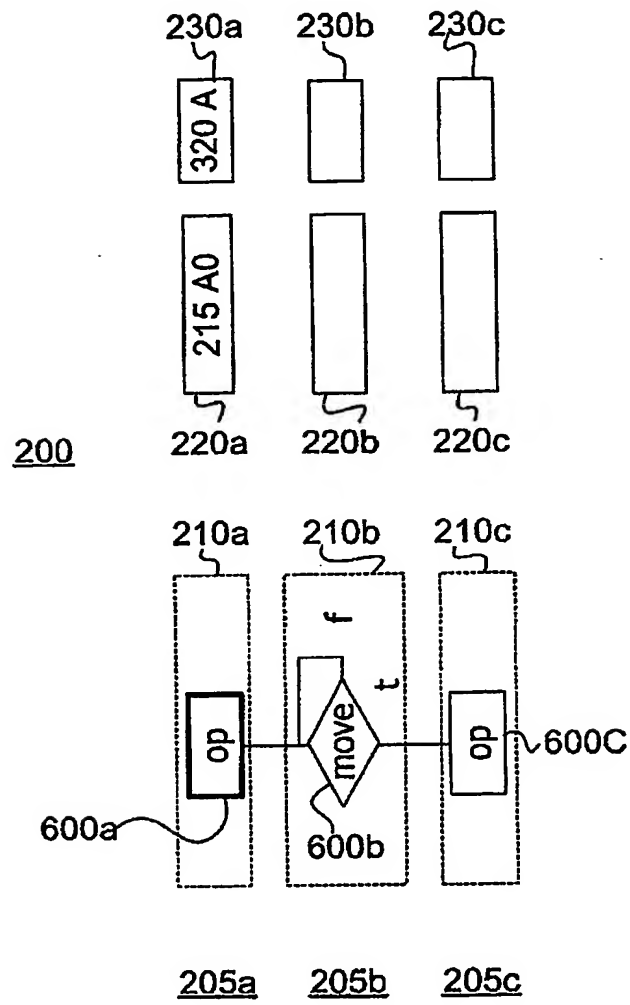


Fig. 6b

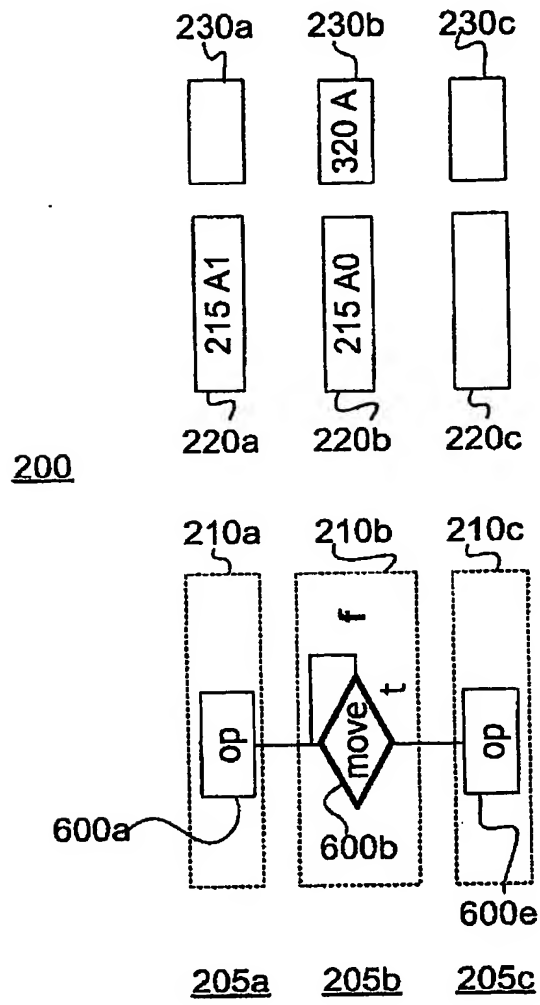


Fig. 6c

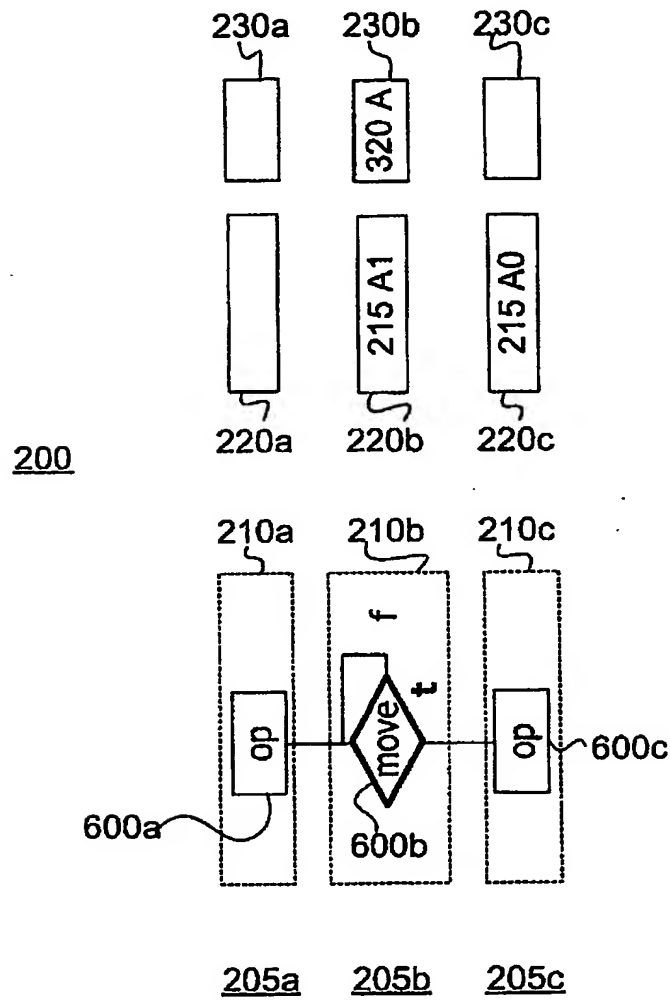
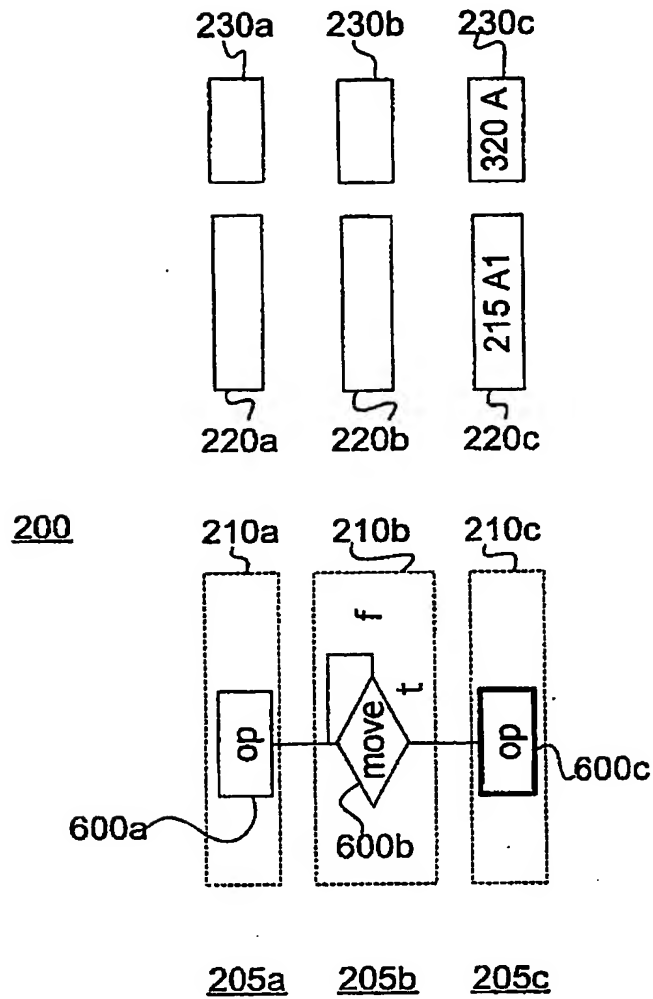


Fig. 6d



0201020-3

0201020-3

Fig. 7

